GREEDY MATCHING IN BIPARTITE RANDOM GRAPHS

NICK ARNOSTI, COLUMBIA UNIVERSITY

ABSTRACT. This paper studies the performance of greedy matching algorithms on bipartite graphs $G = (\mathcal{J}, \mathcal{D}, \mathcal{E})$ in which the edge set $\mathcal{E}$ is random. The main result is that sequentially matching random vertices in $\mathcal{J}$ to random neighbors results in a larger matching than sequentially adding random edges from $\mathcal{E}$. In contrast to previous asymptotic results, this holds for finite graphs with arbitrarily many vertices, and with arbitrary degree distributions for vertices in $\mathcal{J}$. Furthermore, it applies in settings where vertices in $\mathcal{D}$ have arbitrary capacities, and (with minor modifications) to settings where vertices in $\mathcal{J}$ have different sizes and vertices in $\mathcal{D}$ have different expected degree.

The second result is that when each vertex in $\mathcal{D}$ can be assigned to only one vertex in $\mathcal{J}$, then any two vertex-based greedy matching algorithms that consider the vertices of $\mathcal{J}$ in the same order lead to the same probability of matching for each vertex in $\mathcal{J}$. In particular, this implies that the well-known RANKING algorithm (which matches vertices in $\mathcal{J}$ to the highest-ranked feasible vertex in $\mathcal{D}$) has identical average-case performance to the algorithm that matches vertices in $\mathcal{J}$ randomly, despite offering a better worst-case guarantee. By our main result, this also implies that RANKING outperforms the algorithm that sequentially adds random edges.

1. INTRODUCTION

Bipartite matching is a classic problem in the fields of operations research and computer science. Its applications in both the digital and physical worlds are so numerous that it is hopeless to try to enumerate them. For example, companies may need to schedule construction jobs for particular days, or assign programming tasks to developers. Park agencies award hiking permits to interested applicants. Advertising platforms match individual impressions to advertisers. Operating systems assign information to spaces in memory. Conferences assign referees to submitted papers, and later, time slots and conference rooms to accepted papers. In all of these applications, constraints determine which assignments are feasible.

In this paper, we adopt the terminology of assigning a set of jobs $\mathcal{J}$ to a set of days $\mathcal{D}$. The set of possible edges is $E(\mathcal{J}, \mathcal{D}) = \{(j, d) : j \in \mathcal{J}, d \in \mathcal{D}\}$. The edge set $\mathcal{E} \subseteq E(\mathcal{J}, \mathcal{D})$ gives *scheduling* constraints: $(j, d) \in \mathcal{E}$ if and only if job j can be scheduled for day d. The vector $\mathbf{C} \in \mathbb{N}^{\mathcal{D}}$ gives *capacity* constraints: C_d is the maximum number of jobs that can be scheduled on day d.

Definition 1. *Fix $G = (\mathcal{J}, \mathcal{D}, \mathcal{E})$, and $\mathbf{C} \in \mathbb{N}^{\mathcal{D}}$.*

- *A* **matching** *is a set of edges $\mathcal{M} \subseteq E(\mathcal{J}, \mathcal{D})$.*
- *A matching $\mathcal{M}$ is* **feasible** *(with respect to $\mathcal{E}$ and $\mathbf{C}$) if $\mathcal{M} \subseteq \mathcal{E}$, and in the graph $(\mathcal{J}, \mathcal{D}, \mathcal{M})$, each $j \in \mathcal{J}$ has at most one neighbor and each $d \in \mathcal{D}$ has at most C_d neighbors.*
- *A matching $\mathcal{M}$ is* **maximal** *(with respect to $\mathcal{E}$ and $\mathbf{C}$) if it is feasible, and for all $e \in \mathcal{E} \backslash \mathcal{M}$, $\mathcal{M} \cup \{e\}$ is not feasible.*

1.1. **Greedy Matching Algorithms.** This paper considers greedy algorithms, which start with an empty matching, and gradually add edges until reaching a maximal matching. Each of the algorithms studied in this paper is a special case of the following greedy procedure:

1. Select a complete order $\succ^E$ on $E(\mathcal{J}, \mathcal{D})$. Initialize $\mathcal{M} = \emptyset$.
2. In the order $\succ^E$, sequentially add edges to $\mathcal{M}$ so long as the result remains feasible.

Let $\mathcal{M}(\mathcal{E}, \succ^E)$ denote the resulting matching. Different edge orders $\succ^E$ correspond to different matching procedures. This paper considers several simple ways to generate the order $\succ^E$:

EDGE : The order $\succ^E$ is drawn uniformly at random.

SD($\succ^{\mathcal{J}}, \succ^{\mathcal{D}}$): Given an order $\succ^{\mathcal{J}}$ on $\mathcal{J}$ and a collection of orders $\succ^{\mathcal{D}} = \{\succ^{\mathcal{D}}_j\}_{j \in \mathcal{J}}$ on $\mathcal{D}$, the order $\succ^E$ satisfies

$$(j, d) \succ^E (j', d') \Leftrightarrow j \succ^{\mathcal{J}} j' \text{ or } j = j' \text{ and } d \succ^{\mathcal{D}}_j d'.$$

The letters SD stand for "serial dictatorship" – terminology adopted by the matching literature because the jobs sequentially dictate which day they will be assigned to. Serial dictatorships are a simple way to encode priorities among jobs. Two special (randomized) cases of SD are as follows.

VERTEX : Select the order $\succ^{\mathcal{J}}$ and the orders $\succ^{\mathcal{D}}_j$ independently and uniformly at random.

RANKING : Select the order $\succ^{\mathcal{J}}$ uniformly at random. Select an order $\succ$ on $\mathcal{D}$ uniformly at random, and set $\succ^{\mathcal{D}}_j = \succ$ for all $j \in \mathcal{J}$.

These greedy algorithms arise naturally in applications. Below, we briefly discuss their emergence in dynamic matching, static one-sided matching, and two-sided matching.

In many settings, items are allocated dynamically. For example, many wilderness areas have a limited number of camping permits for each day, and prospective campers can select among days for which permits remain. This first-come, first served system corresponds to a serial dictatorship in which $\succ^{\mathcal{J}}$ encodes the order of arrival.

When allocating items in a single round, one natural approach is a "fair" implementation of a serial dictatorship in which the order $\succ^{\mathcal{J}}$ is selected uniformly at random. This is in fact how permits for the popular Half Dome hike in Yosemite are allocated – see `https://www.nps.gov/yose/planyourvisit/hdpermits.htm`. Each applicant can list (and rank) up to seven feasible days. Applicants are ordered by a random lottery, and sequentially assigned to their most preferred day with available permits. The algorithms VERTEX and RANKING can be thought of as a random serial dictatorship in which applicants have independent or identical preferences over days, respectively.

Meanwhile, the procedure EDGE arises if a new lottery number is assigned to each *application* rather than each *applicant*. For example, the website `lottery.broadwaydirect.com` allows users to apply daily for lotteries to win discounted tickets to several broadway musicals. The system randomly selects winners, while assuring that nobody wins tickets to concurrent shows. Although the website does not specify its lottery procedure, the two most likely procedures are to use one lottery for each show (EDGE) or hold a lottery among all applicants (VERTEX).

In two-sided matching contexts, both sides have preferences. A matching is called *stable* if no pair of agents prefer each other to their assigned match partner. Under certain assumptions on preferences, there is a unique stable match, and it can be found by a greedy procedure. For

example, if each potential match between employer and applicant has a "match quality", and both sides prefer matches of higher quality, then the unique stable match is found by a greedy procedure in which $\succ^E$ lists edges in order of their quality. Meanwhile, if employers all agree on a ranking of applicants, then the unique stable match can be found by a serial dictatorship, with high-quality applicants selecting first.

1.2. **Results.** Simple examples show that each of EDGE, VERTEX, and RANKING may outperform the others for particular graphs – see Figure 1. This paper considers their performance when the scheduling constraints $\mathcal{E}$ are random, as defined below.

Definition 2. *Fix $\mathcal{J}, \mathcal{D}$, and $\bar{\mathbf{N}} = \mathbb{N}^{\mathcal{D}}$, where each $\bar{N}_j \in \{0, \dots, |\mathcal{D}|\}$. Let $\psi(\bar{\mathbf{N}})$ be the distribution over subsets of $E(\mathcal{J}, \mathcal{D})$ generated by having each $j \in \mathcal{J}$ independently select $\bar{N}_j$ neighbors in $\mathcal{D}$ uniformly at random.*

Our main result is to show that for this family of random graphs, the size of the matching generated by VERTEX stochastically dominates the size of the matching generated by EDGE.

Theorem 1. *Fix $\mathcal{J}, \mathcal{D}, \bar{\mathbf{N}}, \mathbf{C}$. If $\mathcal{E}$ is drawn from $\psi(\bar{\mathbf{N}})$, then for all $k \in \mathbb{N}$,*

$$\mathbb{P}(|\mathcal{M}(\mathcal{E}, \text{VERTEX})| \geq k) \geq \mathbb{P}(|\mathcal{M}(\mathcal{E}, \text{EDGE})| \geq k).$$

Sections 4.1 and 4.2 establish that this result continues to hold in extensions of the model where jobs have different "sizes" and some days are more likely to be feasible than others.

Section 4.3 shows that when only one job can be scheduled on each day, $\text{SD}(\succ^{\mathcal{J}}, \succ^{\mathcal{D}})$ and $\text{SD}(\succ^{\mathcal{J}}, \tilde{\succ}^{D})$ have identical performance for any $\succ^{\mathcal{J}}$, $\succ^{\mathcal{D}}$ and $\tilde{\succ}^{D}$. In other words, the distribution of matching size is determined by the randomness in the scheduling constraints $\mathcal{E}$, and not by the preferences of individual jobs. With some abuse of notation, we write $j \in \mathcal{M}$ as shorthand for "$(j, d) \in \mathcal{M}$ for some $d \in \mathcal{D}$."

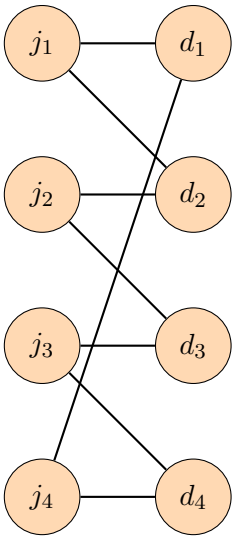
Theorem 2. *Fix $\mathcal{J}, \mathcal{D}, \bar{\mathbf{N}}$. Let $\succ^{\mathcal{J}}$ be an order on $\mathcal{J}$, and let $\succ^{\mathcal{D}}$ and $\tilde{\succ}^{D}$ be two profiles of orders on $\mathcal{D}$. If $C_d = 1$ for all $d \in \mathcal{D}$, and G is drawn from $\psi(\bar{\mathbf{N}})$, then for all $j \in \mathcal{J}$,*

$$\mathbb{P}(j \in \mathcal{M}(\mathcal{E}, \text{SD}(\succ^{\mathcal{J}}, \succ^{\mathcal{D}}))) = \mathbb{P}(j \in \mathcal{M}(\mathcal{E}, \text{SD}(\succ^{\mathcal{J}}, \tilde{\succ}^{D}))).$$

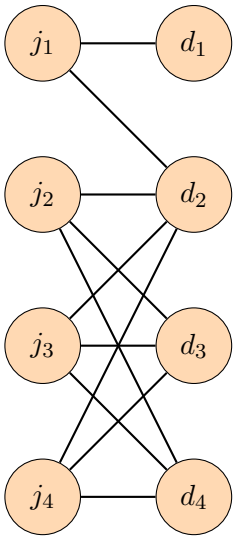
A corollary of Theorem 2 is that when only one job can be scheduled on each day, RANKING and VERTEX have identical performance. By Theorem 1, both result in a stochastically larger matching than EDGE.

2. Related Work

There is a large literature on bipartite matching. Recent applications include the development and analysis of sophisticated matching algorithms for online advertising (Mehta et al. (2007), Goel and Mehta (2008), Feldman et al. (2009), Manshadi et al. (2012)) and Cuckoo Hashing (Dietzfelbinger et al. (2010), Fountoulakis and Panagiotou (2012), Frieze et al. (2018)). The discussion below focuses exclusively on papers that study the greedy algorithms discussed in Section 1.1.



(A) Expected unassigned jobs:
$11/24 \approx 0.46$ under VERTEX,
$8/15 \approx 0.53$ under EDGE, and
$13/24 \approx 0.54$ under RANKING.



(B) Expected unassigned jobs:
$1/4 = 0.25$ under VERTEX,
$8/33 \approx 0.24$ under EDGE, and
$5/24 \approx 0.21$ under RANKING.

FIGURE 1. Pairwise comparisons between VERTEX, EDGE, RANKING. In 1a, VERTEX leads to the most matches, and RANKING the fewest. In 1b, RANKING leads to the most matches, and VERTEX the fewest.

2.1. **Comparisons Between VERTEX and EDGE.** Given that no universal comparison is possible, it is natural to compare these algorithms either according to their worst-case performance or according to their average-case performance on some family of graphs.

Dyer and Frieze (1991) perform a worst-case analysis of EDGE (which they name "GREEDY"), and show that for any $\epsilon > 0$, there exist graphs for which a complete matching is possible, yet EDGE expects to match fewer than $1/2 + \epsilon$ of the vertices. Aronson et al. (1995) perform a worst-case analysis of VERTEX (which they refer to as "Modified Random Greedy" or MRG), and show that on any graph, the ratio of the expected performance of VERTEX to the size of the maximum matching is at least $1/2 + 2.5 \times 10^{-6}$. Together, these papers imply that VERTEX has a better worst-case approximation ratio than EDGE, although the gap may be small.

Dyer et al. (1993) study the asymptotic performance of EDGE and VERTEX (which they refer to as "GREEDY" and "MODIFIED GREEDY," respectively) on two families: sparse Erdös-Renyi random graphs, and random labeled trees. For both families, they show that as the number of vertices grows large, the expected matching size is larger under VERTEX than under EDGE. Although our results are technically incomparable because the present paper studies bipartite random graphs, Theorem 1 is "stronger" than this result in several crucial ways:

- It holds for graphs of arbitrary size, rather than only asymptotically.
- It allows for an arbitrary degree distribution among vertices in $\mathcal{J}$, whereas degrees in Erdös-Renyi random graphs follow a Poisson distribution.
- It allows vertices in $\mathcal{D}$ to have arbitrary (non-identical) capacities, rather than assuming $C_d = 1 \; \forall d$.

- As discussed in Section 4, minor modifications to the proof of Theorem 1 can accommodate heterogeneity in the "size" of vertices in $\mathcal{J}$ and probability of feasibility for vertices in $\mathcal{D}$.

2.2. **The RANKING Algorithm.** Karp et al. (1990) consider a bipartite matching setting in which the edge set $\mathcal{E}$ and order $\succ^{\mathcal{J}}$ are adversarial. They consider two algorithms which they refer to as RANDOM and RANKING, which correspond to SD($\succ^{\mathcal{J}}, \succ^{\mathcal{D}}$) when the orders $\succ^{\mathcal{D}}_j$ are drawn uniformly at random and are (respectively) independently drawn or identical. They show that for any $\mathcal{E}$ and $\succ^{\mathcal{J}}$, the expected size of the matching generated by RANKING is at least $1-1/e$ of the maximum matching size. By contrast, the best such guarantee for the algorithm RANDOM is $1/2$.

More recent work has focused on the case where the order $\succ^{\mathcal{J}}$ is random rather than adversarial – that is, the algorithm that this paper refers to as RANKING. Mahdian and Yan (2011) show that for all edge sets $\mathcal{E}$, the expected size of the matching generated by RANKING is at least 0.696 times the size of the maximum matching. This guarantee is notably larger than the $1/2+2.5\times10^{-6}$ proved by Aronson et al. (1995) for VERTEX. Neither of these guarantees is known to be tight, although the guarantee for RANKING is "nearly" so: Karande et al. (2011) provide a family of instances for which the matching generated by RANKING has size only 0.727 times the size of the maximum matching.

Better worst-case guarantees do not, however imply better average-case performance. Indeed, Lemma 4 in Mastin and Jaillet (2013) notes that on Erdös-Renyi random graphs, the performances of RANKING and VERTEX are identical. Theorem 2 can be seen as a generalization of Lemma 4 from Mastin and Jaillet (2013) along two dimensions:

- It allows an arbitrary degree distribution, rather than assuming an Erdös-Renyi structure.
- It holds for arbitrary deterministic orders $\succ^{\mathcal{J}}$ and $\succ^{\mathcal{D}}$, rather than only when $\succ^{\mathcal{J}}$ and $\succ^{\mathcal{D}}_j$ are drawn uniformly at random.

3. Proof of Theorem 1

The key intuition underlying the proof of Theorem 1 is that it is best to start by scheduling jobs with few feasible days, because jobs with many feasible days can likely be scheduled later. EDGE does the *opposite*: at each step, jobs with more feasible days are more likely to be scheduled. Meanwhile, VERTEX schedules jobs in a random order, and thus is more likely than EDGE to schedule low-degree jobs.

To formalize this idea, Section 3.1 describes Markov chains corresponding to procedures EDGE and VERTEX. In each Markov chain, the matching $\mathcal{M}$ is constructed iteratively, starting with $\mathcal{M}_0 = \emptyset$ and adding one edge at a time. Rather than revealing the random edges $\mathcal{E}$ initially, these chains start with $\mathcal{E}_0 = \emptyset$ and reveal an edge $e = (j, d) \in \mathcal{E}$ only when either i) j is matched, or ii) d reaches its capacity. Potential edges between unassigned jobs and days with excess capacity remain unobserved. The size of the matching generated by each procedure corresponds to the number of steps before the full edge set $\mathcal{E}$ is revealed (implying that no more matches can be formed).

Section 3.2 couples these chains such that the number of unrevealed edges under VERTEX is always weakly larger than the corresponding number under EDGE. Whenever the number of jobs with at least k unrevealed edges is identical in the two chains, the coupling ensures that if the next

job assigned under VERTEX has at least k unrevealed edges, then so does the next job assigned under EDGE. Figure 2 provides an illustration of each chain, and the coupling between them.

3.1. **Proof Step 1: Markov Chain Description.** For both VERTEX and EDGE, we start with an empty edge set and matching $\mathcal{E}_0 = \mathcal{M}_0 = \emptyset$. At each step t, we add one edge between an unscheduled job and a day with idle capacity to the matching $\mathcal{M}_t$ (if no such edge exists, then our procedure has finished). We also reveal any edges involving matched jobs or days with no remaining capacity and add these to the edge set $\mathcal{E}_t$. This ensures that unrevealed edges are precisely those that can be feasibly added to $\mathcal{M}_t$. In other words, for any $e = (j, d)$,

$$\mathcal{M}_t \cup \{e\} \text{ is feasible} \Leftrightarrow e \in \mathcal{E} \backslash \mathcal{E}_t.$$

We now describe the procedures by which edges are added to $\mathcal{M}_t$ under VERTEX and EDGE. We first offer a description in English, and then provide corresponding mathematical expressions. For each procedure, step $t+1$ consists of four stages:

V1 Reveal the job j_{t+1} which ranks highest (according to $\succ^{\mathcal{J}}$) among unassigned jobs for which at least one feasible day has idle capacity.
V2 Observe the days D_{t+1} that are feasible for j_{t+1} and have idle capacity.
V3 Assign j_{t+1} to a random day $d_{t+1} \in D_{t+1}$.
V4 If day d_{t+1} no longer has idle capacity, then for each unassigned job j, observe whether d_{t+1} was feasible for j.

E1 Reveal the job j_{t+1} which has the highest-ranking edge in $\mathcal{E} \backslash \mathcal{E}_t$ (according to $\succ^{E}$).
E2 Observe the days D_{t+1} that are feasible for j_{t+1} and have idle capacity.
E3 Assign j_{t+1} to a random day $d_{t+1} \in D_{t+1}$.
E4 If day d_{t+1} no longer has idle capacity, then for each unassigned job j, observe whether d_{t+1} was feasible for j.

We now provide equations corresponding to each of these chains. Fix a number of feasible days $\bar{N}_j$ for each job j and capacities C_d for each day d. Given a set of edges $\mathcal{E}_t$ and a feasible matching $\mathcal{M}_t \subseteq \mathcal{E}_t$, define

$$\begin{aligned} I_d(\mathcal{M}_t) &= C_d - |\{e \in \mathcal{M}_t : e = (j,d) \text{ for some } j\}| && \text{Idle capacity on day } d. \\ N_j(\mathcal{E}_t) &= \bar{N}_j - |\{e \in \mathcal{E}_t : e = (j,d) \text{ for some } d\}| && \text{Number of unknown feasible days for job } j. \\ F_k(\mathcal{E}_t) &= |\{j : N_j(\mathcal{E}_t) = k\}| && \text{Number of jobs with } k \text{ unknown feasible days.} \end{aligned}$$

The stages V1-V4 above can be expressed as follows:

V1 **Reveal the next job to be assigned.** Select j_{t+1} with

$$\mathbb{P}(j_{t+1} = j | \mathcal{M}_t, \mathcal{E}_t) = \frac{\mathbf{1}(N_j(\mathcal{E}_t) > 0))}{\sum_{j'} \mathbf{1}(N_{j'}(\mathcal{E}_t) > 0)}. \tag{1}$$

V2 **Reveal feasible days for j_{t+1}.** Select $D_{t+1} \subseteq \{d : I_d(\mathcal{M}_t) > 0\}$ uniformly at random among subsets of size $N_{j_{t+1}}(\mathcal{M}_t)$, and define

$$\mathcal{A}_{t+1} = \{(j_{t+1}, d) : d \in D_{t+1}\}. \tag{2}$$

V3 **Assign j_{t+1} to a feasible day.** Select d_{t+1} uniformly at random from D_{t+1}, and set

$$\mathcal{M}_{t+1} = \mathcal{M}_t \cup (j_{t+1}, d_{t+1}). \tag{3}$$

Note that steps V2 and V3 imply that d_{t+1} is selected uniformly at random from days with idle capacity. That is,

$$\mathbb{P}(d_{t+1} = d|\mathcal{M}_t, \mathcal{E}_t, j_{t+1}) = \frac{\mathbf{1}(I_d(\mathcal{M}_t) > 0)}{|\{d : I_d(\mathcal{M}_t) > 0\}|}. \tag{4}$$

V4 **If d_{t+1} has no remaining capacity, reveal neighbors of d_{t+1} in $\mathcal{E}$.** Draw independent binary random variables $\{B_{(t+1)j}\}_{j\in\mathcal{J}}$, with

$$\mathbb{P}(B_{(t+1)j} = 1|\mathcal{E}_t, \mathcal{M}_t, j_{t+1}, D_{t+1}, d_{t+1}) = \frac{N_j(\mathcal{E}_t)}{|\{d : I_d(\mathcal{M}_t) > 0\}|}\mathbf{1}(j \neq j_{t+1})\mathbf{1}(I_{d_{t+1}}(\mathcal{M}_{t+1}) = 0). \tag{5}$$

Set

$$\mathcal{B}_{t+1} = \{(j, d_{t+1}) : B_{(t+1)j} = 1\} \tag{6}$$

$$\mathcal{E}_{t+1} = \mathcal{E}_t \cup \mathcal{A}_{t+1} \cup \mathcal{B}_{t+1}. \tag{7}$$

The stage E1 corresponds to the following:

E1 Select j_{t+1} with

$$\mathbb{P}(j_{t+1} = j|\mathcal{M}_t, \mathcal{E}_t) = \frac{N_j(\mathcal{E}_t)}{\sum_{j'} N_{j'}(\mathcal{E}_t)}. \tag{8}$$

Meanwhile, stages E2-E4 are identical to stages V2-V4. In other words, the only difference between the procedures is that step V1 selects an unassigned job uniformly at random among those with at least one feasible day remaining, whereas step E1 selects among these jobs *in proportion to the number of feasible days remaining.*

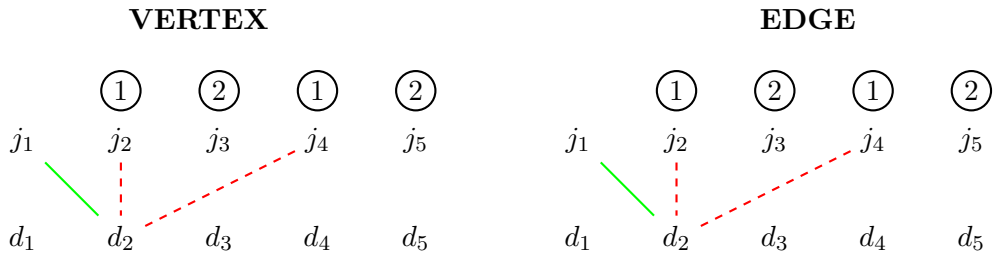
Both procedures terminate when no more edges can be feasibly added – that is, when $F_k(\mathcal{E}_t) = 0$ for all $k \geq 1$.

3.2. **Proof Step 2: Coupling.** In what follows, we use the superscript V to denote the chain corresponding to VERTEX, and E to denote the chain corresponding to EDGE. In other words, $\mathcal{M}_t^V, \mathcal{E}_t^V$ refer to the (random) edges and matching generated after t steps of V1-V4, and $\mathcal{M}_t^E, \mathcal{E}_t^E$ to the (random) edges and matching generated after t steps of E1-E4. For $X \in \{V, E\}$, let j_{t+1}^X be the job assigned at step $t+1$, let D_{t+1}^X be the days to which this job could feasibly be assigned, and let d_{t+1}^X be the day to which this job is actually assigned. Let τ^X be the time satisfying $F_k(\mathcal{E}_{\tau^X}^X) = 0$ for all $k \geq 1$. In other words, τ^X is the time at which it is no longer possible to add feasible edges to $\mathcal{M}_t^X$. The following Lemma immediately implies Theorem 1.

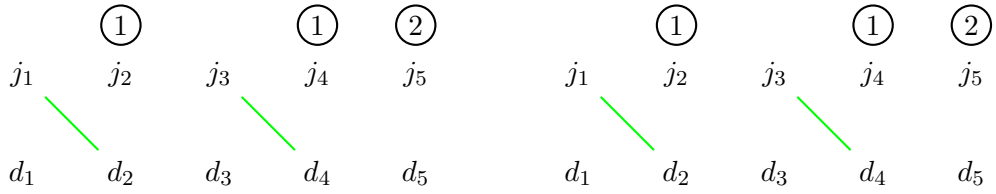
Lemma 1. *There exists a coupling of $(\mathcal{M}_t^V, \mathcal{E}_t^V)$ and $(\mathcal{M}_t^E, \mathcal{E}_t^E)$ such that for all $t \leq \tau^E$ and all $k \in \{1, \ldots, |\mathcal{D}|\}$ the following hold:*

$$I(\mathcal{M}_t^V) = I(\mathcal{M}_t^E). \tag{9}$$

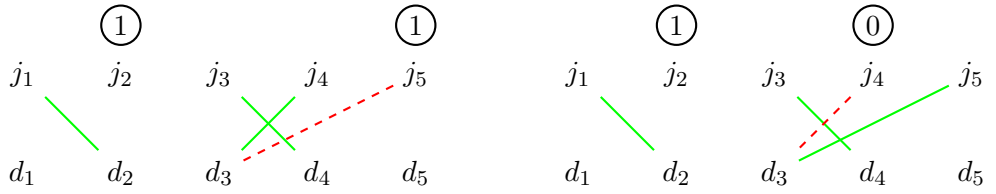
$$\sum_{i\geq k} F_i(\mathcal{E}_t^V) \geq \sum_{i\geq k} F_i(\mathcal{E}_t^E). \tag{10}$$



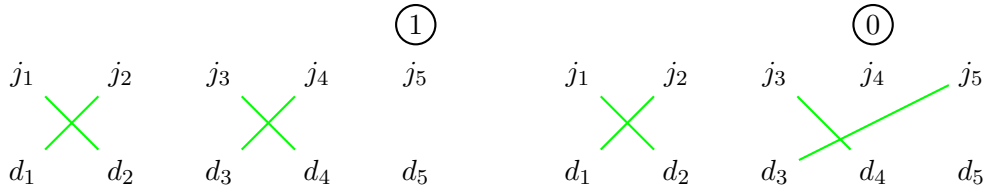
A match is formed between j_1 and d_2. Throughout, (9) ensures that the set of assigned days is identical for the two chains. Because (13) is tight, the coupling ensures that the revealed (red dashed) edges for the two chains coincide. It is revealed that d_2 was also feasible for j_2 and j_4.



The next match under VERTEX involves j_3, for which two remaining days are feasible. Because (10) is tight, the coupling ensures that the next match under EDGE also involves a job for which two remaining days are feasible. Because (13) is tight, the (empty) sets of revealed edges coincide. Jobs j_2 and j_4 are more likely to go unassigned than j_5, for which two of $\{d_1, d_3, d_5\}$ are feasible.



Because (10) is tight, the next job assigned under EDGE has weakly more remaining feasible days (see Lemma 2). The next match under VERTEX involves j_4, whereas under EDGE it involves j_5. For the first time, ϕ is not the identity: it permutes j_4 and j_5. Under EDGE, it is revealed that d_3 was feasible for j_4, which can no longer be feasibly assigned.



EDGE must match j_2. VERTEX randomly selects which of j_2 and j_5 to match. Job j_2 is assigned to d_1, and it is revealed that d_1 was not feasible for j_5. EDGE has completed, whereas VERTEX will continue for one more step.

FIGURE 2. A visualization of the chains corresponding to VERTEX and EDGE on an instance with five jobs and five days. Each job has two feasible days, and only one job can be scheduled for each day. In each step, an unscheduled job is assigned to one of its remaining feasible days (shown by solid green lines), and the set of other jobs that were feasible for this day is revealed (shown with dashed red lines). The numbers above each job denote the number of remaining feasible days for this job. Under VERTEX, jobs are selected in uniformly at random, whereas under EDGE, they are selected in proportion to the number of remaining feasible days.

Proof. First, (4) implies that if $I(\mathcal{M}_t^V) = I(\mathcal{M}_t^E)$, then regardless of the realizations j_{t+1}^V, j_{t+1}^E, the chains can be coupled so that $d_{t+1}^V = d_{t+1}^E$, and therefore (3) implies $I(\mathcal{M}_{t+1}^V) = I(\mathcal{M}_{t+1}^E)$. In light of this fact, we write d_{t+1} in place of $d_{t+1}^V = d_{t+1}^E$, and $I(\mathcal{M}_t)$ in place of $I(\mathcal{M}_t^V) = I(\mathcal{M}_t^E)$.

We now turn to (10). For $X \in \{V, E\}$ we define

$$\mathcal{A}_{t+1}^X = \{(j_{t+1}^X, d) : d \in D_{t+1}^X\} \tag{11}$$

Lemma 2 states that it is possible to couple $\mathcal{A}_{t+1}^V, \mathcal{A}_{t+1}^E$ such that

$$\sum_{i \geq k} F_i(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V) \geq \sum_{i \geq k} F_i(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^V). \tag{12}$$

That is, an analog of (10) holds after stages V3 and E3. If $I_{d_{t+1}}(\mathcal{M}_{t+1}) > 0$, then by (5) and (6) $\mathcal{B}_{t+1}^V = \mathcal{B}_{t+1}^E = \emptyset$, so for $X \in \{V, E\}$, (7) implies that $\mathcal{E}_{t+1}^X = \mathcal{E}_t^X \cup \mathcal{A}_{t+1}^X$, and there is nothing more to do.

Otherwise, we must couple B_{t+1}^V and B_{t+1}^E. It follows from (12) that there exists a permutation ϕ on $\mathcal{J}$ such that $\phi(j_{t+1}^V) = j_{t+1}^E$, and for all j,

$$N_j(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V) \geq N_{\phi(j)}(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^E). \tag{13}$$

For each j, if equality holds in (13), couple $B_{(t+1)j}^V$ and $B_{(t+1)\phi(j)}^E$ such that $B_{(t+1)j}^V = B_{(t+1)\phi(j)}^E$. If the inequality in (13) is strict, generate $B_{(t+1)j}^V$ and $B_{(t+1)\phi(j)}^E$ independently. This ensures that

$$N_j(\mathcal{E}_{t+1}^V) = N_j(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V \cup \mathcal{B}_{t+1}^V) \geq N_{\phi(j)}(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^E \cup \mathcal{B}_{t+1}^E) = N_{\phi(j)}(\mathcal{E}_{t+1}^E) \tag{14}$$

for all j, and therefore that (10) holds.

□

Lemma 2. *If* (10) *holds, then it is possible to couple* $\mathcal{A}_{t+1}^V, \mathcal{A}_{t+1}^E$ *such that for* $k \in \{1, \ldots, |\mathcal{D}|\}$,

$$\sum_{i \geq k} F_i(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V) \geq \sum_{i \geq k} F_i(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^E).$$

Proof. For $X \in \{V, E\}$ define $K_{t+1}^X = N_{j_{t+1}}(\mathcal{E}_t^X)$ to be the number of feasible days for job j_{t+1} that still have idle capacity. Then the definition of $\mathcal{A}_{t+1}^X$ in (2) implies that for $k \in \{1, \ldots, |\mathcal{D}|\}$,

$$\sum_{i \geq k} F_i(\mathcal{E}_t^X \cup \mathcal{A}_{t+1}^X) = \sum_{i \geq k} F_i(\mathcal{E}_t^X) - \mathbf{1}(K_{t+1}^X \geq k).$$

Thus, it is enough to show that if $\sum_{i \geq k} F_i(\mathcal{E}_t^V) = \sum_{i \geq k} F_i(\mathcal{E}_t^E)$, it is possible to couple K_{t+1}^V and K_{t+1}^E such that $K_{t+1}^V \geq k$ implies that $K_{t+1}^E \geq k$. In other words, we must show that

$$\mathbb{P}(K_{t+1}^E \geq k | \mathcal{M}_t^E, \mathcal{E}_t^E) \geq \mathbb{P}(K_{t+1}^V \geq k | \mathcal{M}_t^V, \mathcal{E}_t^V).$$

Note that (1) implies that for $i \geq 1$,

$$\mathbb{P}(K_{t+1}^V = i | \mathcal{M}_t^V, \mathcal{E}_t^V) = \frac{F_i(\mathcal{E}_t^V)}{\sum_{j \geq 1} F_j(\mathcal{E}_t^V)}.$$

Analogously, (8) implies that

$$\mathbb{P}(K_{t+1}^E = i | \mathcal{M}_t^E, \mathcal{E}_t^E) = \frac{i F_i(\mathcal{E}_t^E)}{\sum_{j \geq 1} j F_j(\mathcal{E}_t^E)}.$$

It follows that if $\sum_{i\geq k} F_i(\mathcal{E}_t^V) = \sum_{i\geq k} F_i(\mathcal{E}_t^E)$, then

$$\begin{aligned}
\mathbb{P}(K_{t+1}^V \geq k | \mathcal{M}_t^V, \mathcal{E}_t^V) &= \frac{\sum_{i\geq k} F_i(\mathcal{E}_t^V)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^V)} \\
&\leq \frac{\sum_{i\geq k} F_i(\mathcal{E}_t^V)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^E)} \\
&= \frac{\sum_{i\geq k} F_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^E)} \\
&\leq \frac{\sum_{i\geq k} iF_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} iF_i(\mathcal{E}_t^E)} \\
&= \mathbb{P}(K_{t+1}^E \geq k | \mathcal{M}_t^E, \mathcal{E}_t^E).
\end{aligned}$$

The first inequality follows from (10), and the final inequality follows because

$$\begin{aligned}
\frac{\sum_{i\geq k} F_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^E)} &= \left(1 + \sum_{i<k} kF_i(\mathcal{E}_t^E) / \sum_{i\geq k} kF_i(\mathcal{E}_t^E)\right)^{-1} \\
&\leq \left(1 + \sum_{i<k} iF_i(\mathcal{E}_t^E) / \sum_{i\geq k} iF_i(\mathcal{E}_t^E)\right)^{-1} = \frac{\sum_{i\geq k} iF_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} iF_i(\mathcal{E}_t^E)}.
\end{aligned}$$

□

4. Extensions

We now discuss extensions of Theorem 1 to more general settings and matching algorithms.

In our first extension, jobs have different "sizes" S_j. For example, C_d might represent the number of hours that developer d can work, and S_j the number of hours that job j will take to complete. Alternately, in the context of hiking permits or theater performances, C_d might represent the number of permits or tickets available on day d, and S_j the number of permits or tickets requested by group j.

In our second extension, some days are more likely than others to be feasible for each job. For example,a programming task may be more likely to be feasible for an experienced developer than an inexperienced one. Analogously, weekends may be feasible for more hikers or theater-goers than weekdays.

In our third extension, we consider other matching algorithms. In particular, we study SD when the orders $\succ_j$ used to assign job j are nonrandom. For example, one might assign each job to the earliest feasible date. Alternately, the lottery for Half Dome permits allows each group to specify preferences over feasible dates, and always assigns groups to their most preferred feasible date with availability.

4.1. Job Sizes.

We add to the primitives $\bar{\mathbf{N}}$ and $\mathbf{C}$ the size vector $\mathbf{S} \in \mathbb{N}^{|\mathcal{D}|}$. We make two key assumptions:

- Size is independent from the number of feasible dates. Formally, the size of job j is $S_{\rho(j)}$, where ρ is a uniform random bijection from $\mathcal{D}$ to $\{1,\dots,|\mathcal{D}|\}$.
- Small amounts of overbooking are allowed. Formally, we consider it feasible to add (j,d) to $\mathcal{M}$ so long as $(j,d)\in\mathcal{E}$ and d has extra capacity under $\mathcal{M}$ (even if $S_{\rho(j)}$ exceeds this excess capacity).

Under these assumptions, an analogous proof holds. In particular, we reveal ρ incrementally: after step t, we have a matching $\mathcal{M}_t$ and an injection $\rho_t : \{j : (j,d)\in\mathcal{M}_t \text{ for some } d\}\to\{1,\dots,|\mathcal{D}|\}$. In steps V1 and E1, we reveal the value $\rho_{t+1}(j_{t+1})$ (that is, the size of the job matched at time $t+1$) and ensure that ρ_{t+1} is consistent with ρ_t on the domain of ρ_t. We define

$$I_d(\mathcal{M}_t,\rho_t) = C_d - \sum_{j':(j',d)\in\mathcal{M}_t} S_{\rho_t(j')},$$

and replace (5) with

$$\mathbb{P}(B_{(t+1)j}=1|\mathcal{E}_t,\mathcal{M}_t,j_{t+1},\rho_{t+1},D_{t+1},d_{t+1}) = \frac{N_j(\mathcal{E}_t)}{|\{d : I_d(\mathcal{M}_t)>0\}|}\mathbf{1}(j\neq j_{t+1})\mathbf{1}(I_{d_{t+1}}(\mathcal{M}_{t+1})\le 0).$$

We then couple the two chains as in Lemma 1. In order to maintain (9), we ensure that for all $t<\tau^E$ we have $d^V_{t+1}=d^E_{t+1}$ (as before) and $\rho^V_{t+1}(j^V_{t+1})=\rho^E_{t+1}(j^E_{t+1})$. That is, the size of the group matched at time t is identical in the two chains. It follows that both the number and total size of matched jobs is higher under VERTEX than EDGE.

4.2. **Non-Uniform Scheduling Constraints.** We add to the primitives $\bar{\mathbf{N}}$ and $\mathbf{C}$ (and, if desired, $\mathbf{S}$) the probability vector $\mathbf{P}\in[0,1]^{\mathcal{D}}$ with $\sum_d P_d = 1$. We make the following key assumption:

- The $\bar{\mathbf{N}}_j$ edges involving job j are drawn independently (with replacement) from $\mathbf{P}$.

Formally, this requires allowing $\mathcal{E}$ to be a multiset – that is, (j,d) may appear in $\mathcal{E}$ multiple times. The stages V2-V4 and E2-E4 are modified as follows.

V2/E2 Determine the multiset $\mathcal{A}_{t+1}=\{(j_{t+1},d_{(t+1)i})\}_{i=1}^{N_j(\mathcal{M}_t)}$ by drawing the $d_{(t+1)i}$ iid, with

$$\mathbb{P}(d_{(t+1)i}=d|\mathcal{M}_t,\mathcal{E}_t,j_{t+1}) = \frac{P_d\mathbf{1}(I_d(\mathcal{M}_t)>0)}{\sum_{d'}P_{d'}\mathbf{1}(I_{d'}(\mathcal{M}_t)>0)}.$$

V3/E3 Choose $e_{t+1}=(j_{t+1},d_{t+1})$ uniformly from $\mathcal{A}_{t+1}$.

V4/E4 As before, if $j=j_{t+1}$ or $I_{d_{t+1}}(\mathcal{M}_{t+1})>0$, then $B_{(t+1)j}=0$. Otherwise, the value $B_{(t+1)j}$ follows a binomial distribution with parameters $N_j(\mathcal{M}_t)$ and $P_{d_{t+1}}/\sum_{d'}P_{d'}\mathbf{1}(I_{d'}(\mathcal{M}_t)>0)$. For $j\in\mathcal{J}$, let $\mathcal{B}_{t+1}$ contain $B_{(t+1)j}$ copies of the edge (j,d_{t+1}).

We then couple the two chains as in Lemma 1. The only necessary modification is to the coupling of $B^V_{(t+1)j}$ and $B^E_{(t+1)j}$ following (13). In this case, draw the values $B^E_{(t+1)j}$ as described in V4/E4 and set

$$B^V_{(t+1)j} = B^E_{(t+1)\phi(j)} + \tilde{B}_{(t+1)j},$$

where $\tilde{B}_{(t+1)j}=0$ if $j=j^V_{t+1}$ or $I_{d^V_{t+1}}(\mathcal{M}^V_{t+1})>0$, and otherwise $\tilde{B}_{(t+1)j}$ follows a binomial distribution with parameters $N_j(\mathcal{E}^V_t)-N_{\phi(j)}(\mathcal{E}^E_t)$ and $P_{d_{t+1}}/\sum_{d'}P_{d'}\mathbf{1}(I_{d'}(\mathcal{M}_t)>0)$. As before, this coupling ensures that

$$N_j(\mathcal{E}^V_t\cup\mathcal{A}^V_{t+1}\cup\mathcal{B}^V_{t+1}) \ge N_{\phi(j)}(\mathcal{E}^E_t\cup\mathcal{A}^E_{t+1}\cup\mathcal{B}^E_{t+1}).$$

4.3. **Equivalence of Serial Dictatorships.** This section proves Theorem 2, which states that when $C_d = 1$ for all d, the probability that a given job j matches under SD($\succ^{\mathcal{J}}\succ^{\mathcal{D}}$) does not depend on $\succ^{\mathcal{D}}$. As before, we iteratively construct $\mathcal{E}_t$ and $\mathcal{M}_t$. Step $t+1$ consists of the following four stages:

SD1 Let j_{t+1} be the first job (according to $\succ^{\mathcal{J}}$) that is unmatched in $\mathcal{M}_t$ and has $N_j(\mathcal{E}_t) > 0$.

SD2 Select $D_{t+1} \subseteq \{d : I_d(\mathcal{M}_t) > 0\}$ uniformly at random among subsets of size $N_{j_{t+1}}(\mathcal{M}_t)$. Define

$$\mathcal{A}_{t+1} = \{(j_{t+1}, d) : d \in D_{t+1}\}.$$

SD3 Let d_{t+1} be the first element in D_{t+1} (according to $\succ_{j_{t+1}}$), and set

$$\mathcal{M}_{t+1} = \mathcal{M}_t \cup (j_{t+1}, d_{t+1}).$$

SD4 Draw independent binary random variables $\{B_{(t+1)j}\}_{j\in\mathcal{J}}$, with

$$\mathbb{P}(B_{(t+1)j} = 1|\mathcal{E}_t, \mathcal{M}_t, D_{t+1}) = \frac{N_j(\mathcal{E}_t)}{|\{d : I_d(\mathcal{M}_t) > 0\}|}\mathbf{1}(j \neq j_{t+1})\mathbf{1}(I_{d_{t+1}}(\mathcal{M}_{t+1}) = 0). \tag{15}$$

Let $\mathcal{B}_{t+1} = \{(j, d_{t+1}) : B_{(t+1)j} = 1\}$, and set

$$\mathcal{E}_{t+1} = \mathcal{E}_t \cup \mathcal{A}_{t+1} \cup \mathcal{B}_{t+1}.$$

Stages SD2 and SD4 are identical to V2 and V4, respectively. The only differences between this Markov chain and that given in V1-V4 are that in steps SD1 and SD3, job j_{t+1} and day d_{t+1} are determined by $\succ^{\mathcal{J}}$ and $\succ_{j_{t+1}}$, and thus are deterministic given $\mathcal{M}_t, \mathcal{E}_t$, and D_{t+1}.

Let $\mathcal{M}_t, \mathcal{E}_t$ be the (random) matching and edge set from SD($\succ^{\mathcal{J}}, \succ^{\mathcal{D}}$) and $\mathcal{M}'_t, \mathcal{E}'_t$ be the corresponding matching and edge set under SD($\succ^{\mathcal{J}}, \tilde{\succ}^{D}$). We claim that the chains can be coupled such that for all j, t,

$$\{j : (j, d) \in \mathcal{M}_t \text{ for some } d\} = \{j : (j, d) \in \mathcal{M}'_t \text{ for some } d\} \tag{16}$$

$$N_j(\mathcal{E}_t) = N_j(\mathcal{E}'_t), \tag{17}$$

from which Theorem 2 follows. If (17) holds at t, it follows immediately from SD1 that

$$j_{t+1} = j'_{t+1}, \tag{18}$$

and therefore (16) holds for $\mathcal{M}_{t+1}$ and $\mathcal{M}'_{t+1}$. Furthermore, because $C_d = 1$ for all d, each round reduces the set of available days by one, so

$$|\{d : I_d(\mathcal{M}_t) > 0\}| = \left|\{d : I_d(\mathcal{M}'_t) > 0\}\right| = |\mathcal{D}| - t. \tag{19}$$

It follows from (15), (17), (18) and (19) that for all j,

$$\mathbb{P}(B_{(t+1)j} = 1|\mathcal{E}_t, \mathcal{M}_t, D_{t+1}) = \mathbb{P}(B'_{(t+1)j} = 1|\mathcal{E}'_t, \mathcal{M}'_t, D'_{t+1}),$$

so we can ensure that $B_{(t+1)j} = B'_{(t+1)j}$, and therefore that $N_j(\mathcal{E}_{t+1}) = N_j(\mathcal{E}'_{t+1})$.

5. Discussion

This paper shows that when scheduling constraints are drawn randomly, sequentially adding random edges yields a smaller matching than sequentially attempting to match random vertices. This

is because selecting random edges is likely to match high-degree vertices first, increasing the risk that low-degree vertices go unmatched. To formalize this simple intuition, I couple carefully chosen Markov chains corresponding to each procedure. Prior work has compared matching algorithms by their worst-case performance, or their asymptotic performance as the size of the graph grows. As far as I know, this is the first paper to provide a comparison of greedy matching algorithms for random graphs of arbitrary size. Additionally, in contrast to prior work, this paper considers many-to-one matching and allows for an arbitrary degree distribution among jobs.

Our findings suggest that in one-sided allocation, running a lottery among applicants should typically result in a larger assignment than generating separate lottery numbers for each application. Although real-world applications often have features not captured by our model, the extensions considered in Sections 4.1 and 4.2 suggest that this conclusion is fairly robust. By contrast, the conclusion of Theorem 2 that VERTEX and RANKING have identical performance is less robust: examples in the Appendix show that this no longer holds if $C_d > 1$ or some days are more likely to be feasible than others.

One natural direction for future research is to understand the magnitude of the difference between the sizes of the matchings produced by VERTEX and EDGE. This will of course depend on the graph parameters. For example, the differences should be small if all vertices have high degree, or if there is a large imbalance between the sides, in which case the number of matches will be close to the minimum capacity of the two sides. One natural model is to draw the number of feasible days N_j for each job and the capacity C_d for each day iid from given distributions, while fixing $|\mathcal{J}| / |\mathcal{D}|$ and taking $|\mathcal{D}| \to \infty$. Analysis of this model could provide insights into the size of the gap between these greedy procedures, and between these procedures and the maximum matching.

References

Aronson, J., Dyer, M., Frieze, A., and Suen, S. (1995). Randomized greedy matching ii. *Random Structures & Algorithms*, 6(1):55–73.

Dietzfelbinger, M., Goerdt, A., Mitzenmacher, M., Montanari, A., Pagh, R., and Rink, M. (2010). Tight thresholds for cuckoo hashing via xorsat. In Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., and Spirakis, P., editors, *Automata, Languages and Programming*, volume 6198 of *Lecture Notes in Computer Science*, pages 213–225. Springer Berlin Heidelberg.

Dyer, M. and Frieze, A. (1991). Randomized greedy matching. *Random Structures & Algorithms*, 2(1):29–45.

Dyer, M., Frieze, A., and Pittel, B. (1993). The average performance of the greedy matching algorithm. *The Annals of Applied Probability*, 3(2):pp. 526–552.

Feldman, J., Mehta, A., Mirrokni, V., and Muthukrishnan, S. (2009). Online stochastic matching: Beating 1-1/e. *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126.

Fountoulakis, N. and Panagiotou, K. (2012). Sharp load thresholds for cuckoo hashing. *Random Structures & Algorithms*, 41(3):306–333.

Frieze, A., Melsted, P., and Mitzenmacher, M. (2018). An analysis of random-walk cuckoo hashing.

Goel, G. and Mehta, A. (2008). Online budgeted matching in random input models with applications to adwords. *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 982–991.

Karande, C., Aranyak, M., and Pushkar, T. (2011). Online bipartite matching with unknown distributions. In *ACM Symposium on Theory of Computing (STOC)*.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. (1990). An optimal algorithm for on-line bipartite matching. *Proceedings of the twenty-second annual ACM symposium on theory of computing*, pages 352–358.

Mahdian, M. and Yan, Q. (2011). Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In *ACM Symposium on Theory of Computing (STOC)*.

Manshadi, V., Oveis Gharan, S., and Saberi, A. (2012). Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573.

Mastin, A. and Jaillet, P. (2013). Greedy online bipartite matching on random graphs. *arXiv preprint arXiv:1307.2536*.

Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. (2007). Adwords and generalized online matching. *J. ACM*, 54(5).

6. Necessity of Conditions of Theorem 2

The following example illustrates that $\text{SD}(\succ^{\mathcal{J}}, \succ^{\mathcal{D}})$ and $\text{SD}(\succ^{\mathcal{J}}, \tilde{\succ}^{\mathcal{D}})$ do not necessarily lead to the same number of matches when $C_d > 1$, even if feasibility is distributed uniformly. To show this, it suffices to show that the performance of VERTEX and RANKING differ. In the following example, under RANKING, schools fill sooner, leaving more later students unmatched.

Example 1. *There are* 2 *days, each with capacity* $|\mathcal{J}|/2$*. One third of jobs consider both days feasible, and two thirds consider only one day feasible, with this day chosen uniformly at random.*

When $|\mathcal{J}| = 4$*, the expected number of unassigned jobs under* VERTEX *is* $1/2$*, and the expected number of unassigned jobs under* RANKING *is* $14/27$*.*

As $|\mathcal{J}| \to \infty$*, the expected fraction of unassigned jobs under* VERTEX *approaches zero, and the expected fraction of unassigned jobs under* RANKING *approaches* $1/12$*.*

The following example illustrates that $\text{SD}(\succ^{\mathcal{J}}, \succ^{\mathcal{D}})$ and $\text{SD}(\succ^{\mathcal{J}}, \tilde{\succ}^{\mathcal{D}})$ do not necessarily lead to the same number of expected matches when feasibility is not uniform, even if $C_d = 1$ for all d.

Example 2. *There are two jobs and two days, with* $C_1 = C_2 = 1$*. Both days are feasible for* j_1*, and only one day is feasible for* j_2*, with* $\mathbb{P}((j_2, d_1) \in \mathcal{E}) = 1 - \mathbb{P}((j_2, d_2) \in \mathcal{E}) = p$*. Job* j_1 *appears first in* $\succ^{\mathcal{J}}$*. If* $\succ^{\mathcal{D}}_{j_1}$ *ranks* d_1 *first, the probability of forming two matches is* $1 - p$*; if* $\succ^{\mathcal{D}}_{j_1}$ *ranks* d_2 *first, this probability is* p*.*

Although in this example, VERTEX and RANKING perform identically, there exist examples with $|\mathcal{J}| = |\mathcal{D}| = 3$ and $C_d = 1$ for all d for which VERTEX and RANKING have different expected performance.